ARMY RESEARCH LABORATORY

# Analysis of Behind-Armor Debris Fragments From Physics-Based Simulations

**by Jerry A. Clarke, Hubert W. Meyer, and Eric R. Mark**

**ARL-CR-590** **February 2007**

**prepared by**

**U.S. Army Research Laboratory**
**Aberdeen Proving Ground, MD 21005**

**and**

**Dynamic Science, Inc.**
**1003 Old Philadelphia Rd., Ste. 210**
**Aberdeen, MD 21001**

**under contract**

**DAAD-1702-C0071**

**Army Research Laboratory**
Aberdeen Proving Ground, MD 21005-5067

**ARL-CR-590** **February 2007**

# Analysis of Behind-Armor Debris Fragments From Physics-Based Simulations

**Jerry A. Clarke and Eric R. Mark**
**Computational and Information Sciences Directorate, ARL**

**Hubert W. Meyer**
**Dynamic Science, Inc.**

**prepared by**

**U.S. Army Research Laboratory**
**Aberdeen Proving Ground, MD 21005**

**and**

**Dynamic Science, Inc.**
**1003 Old Philadelphia Rd., Ste. 210**
**Aberdeen, MD 21001**

**under contract**

**DAAD-1702-C0071**

| **REPORT DOCUMENTATION PAGE** | | *Form Approved* *OMB No. 0704-0188* |
|---|---|---|

| | | |
|---|---|---|
| **1. REPORT DATE** *(DD-MM-YYYY)* February 2007 | **2. REPORT TYPE** Final | **3. DATES COVERED (From - To)** November 2005–August 2006 |
| **4. TITLE AND SUBTITLE** Analysis of Behind-Armor Debris Fragments From Physics-Based Simulations | | **5a. CONTRACT NUMBER** DAAD-1702-C0071 |
| | | **5b. GRANT NUMBER** |
| | | **5c. PROGRAM ELEMENT NUMBER** |
| **6. AUTHOR(S)** Jerry A. Clarke, Hubert W. Meyer,* and Eric R. Mark | | **5d. PROJECT NUMBER** |
| | | **5e. TASK NUMBER** |
| | | **5f. WORK UNIT NUMBER** |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** U.S. Army Research Laboratory, Aberdeen Proving Ground, MD 21005 Dynamic Science, Inc., 1003 Old Philadelphia Rd., Ste. 210, Aberdeen, MD 21001 | | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** U.S. Army Research Laboratory ATTN: AMSRD-ARL-CI-HC Aberdeen Proving Ground, MD 21005-5067 | | **10. SPONSOR/MONITOR'S ACRONYM(S)** |
| | | **11. SPONSOR/MONITOR'S REPORT NUMBER(S)** ARL-CR-590 |

**12. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**13. SUPPLEMENTARY NOTES**
*Dynamic Science, Inc., 1003 Old Philadelphia Rd., Ste. 210, Aberdeen, MD 21001

**14. ABSTRACT**
The multimaterial, large deformation, strong shock wave, solid mechanics code CTH is heavily used for armor penetration applications. It would be extremely beneficial to use this type of physics-based simulation to produce a realistic behind-armor debris field as input to survivability and lethality codes. Utilizing the capabilities of the interdisciplinary computing environment, we have developed a tool that is capable of identifying individual fragments in a CTH flat mesh or adaptive mesh refinement calculation. Additionally, this tool produces an estimate of the volume (thus mass) and velocity of these arbitrarily shaped fragments, outputting the results into a text file that is suitable for use as input to survivability/lethality calculations.

**15. SUBJECT TERMS**
behind-armor debris, physics-based simulation, interdisciplinary computing

| **16. SECURITY CLASSIFICATION OF:** | | | **17. LIMITATION OF ABSTRACT** | **18. NUMBER OF PAGES** | **19a. NAME OF RESPONSIBLE PERSON** Jerry A. Clarke |
|---|---|---|---|---|---|
| **a. REPORT** UNCLASSIFIED | **b. ABSTRACT** UNCLASSIFIED | **c. THIS PAGE** UNCLASSIFIED | UL | 18 | **19b. TELEPHONE NUMBER** (*Include area code*) 410-278-9279 |

**Standard Form 298 (Rev. 8/98)**
**Prescribed by ANSI Std. Z39.18**

## Contents

## List of Figures

## 1. Introduction

Behind-armor debris (BAD) is a major cause of damage in military vehicles that have been perforated by a penetrator, bullet, or fragment. The ability to predict the debris field resulting from attack by such a threat is critical to assessing and improving the survivability of our tactical systems. The U.S. Army Research Laboratory (ARL) has been working to develop the capability to numerically model the BAD resulting from armor perforation.

Modeling of the debris field has historically been done by statistically analyzing data from carefully controlled experiments. The difficulty of collecting this information makes it an expensive and lengthy process. Supplementing these experiments with numerical simulations is a natural synergy, but it has not yet been successfully exploited because previous computer systems were unable to cope with the daunting size of the problems.

With the addition of the latest computers to the ARL Major Shared Resource Center, numerical modeling of these experiments is now within reach. The Eulerian shock physics code CTH was chosen to model the experiment. The experiment, modeled as a demonstration of the technique, consists of a 30-mm armor-piercing discarding sabot round perforating a 1-in-thick armor steel plate. The resulting BAD impacts a large (2- × 2-ft), thin (1/32-in) mild steel witness plate placed 2 ft behind the armor. Perforations made in the witness plate by the debris are measured, and conclusions drawn about the size, mass, spatial distribution, and velocity of the debris field. This is painstaking work, but it results in a reasonably accurate characterization of the debris field.

The difficulty in modeling this experiment arises primarily from two factors. First, the experiment is inherently three-dimensional in nature, and, thus, any simulation of the experiment must be done in three dimensions. The second factor is the wide range of length scales: the 1/32-in witness requires a fine grid resolution that, when extended over the 2-ft air space and large area of the witness, requires one half of a billion cells for a relatively coarse resolution (two cells through the thickness of the witness plate). Compounding the problem, the small cell size requires a small integration time-step, so a huge number of computation cycles are required to fly the debris through the 2-ft airspace.

Extracting the necessary information from the CTH calculation, however, is not a straightforward process. CTH uses structured meshes with cell-centered values for material volume fraction, velocity, pressure, etc. (*1*). The mass and velocity of individual fragments is ultimately useful. Fortunately, the functionality available in the interdisciplinary computing environment (ICE) makes it possible to create a system capable of extracting the necessary information from CTH to be used as input to vulnerability and survivability codes.

## 2. Manipulating CTH Data

Both adaptive mesh refinement (AMR) and non-AMR (flat mesh) can generate output in several proprietary formats. To enable post processing via Spymaster (an internal system to CTH which will be discussed later), CTH can produce data files in a special spyplt format. On parallel platforms, CTH will typically save one spyplt file per processor used in the calculation. Since producing CTH calculations regularly requires tens or hundreds of millions of cells in the computational mesh, the amount of data saved to disk can be enormous. In a flat-mesh calculation, the domain is decomposed one piece per processor. In an AMR calculation, each processor can potentially be assigned many small blocks, which are refined based on some user-defined criteria.

Flat-mesh blocks are assigned to processors so that their interiors overlap by one grid cell. Each processor could potentially have a similar computational block; however, each block may not be exactly the same size. AMR blocks, however, are all topologically identical (e.g., 12 × 12 × 12) but are geometrically refined dynamically based on some user-defined criteria. As shown in figure 1, each processor is assigned one or more blocks representing a section of the computational domain on which it is to operate.

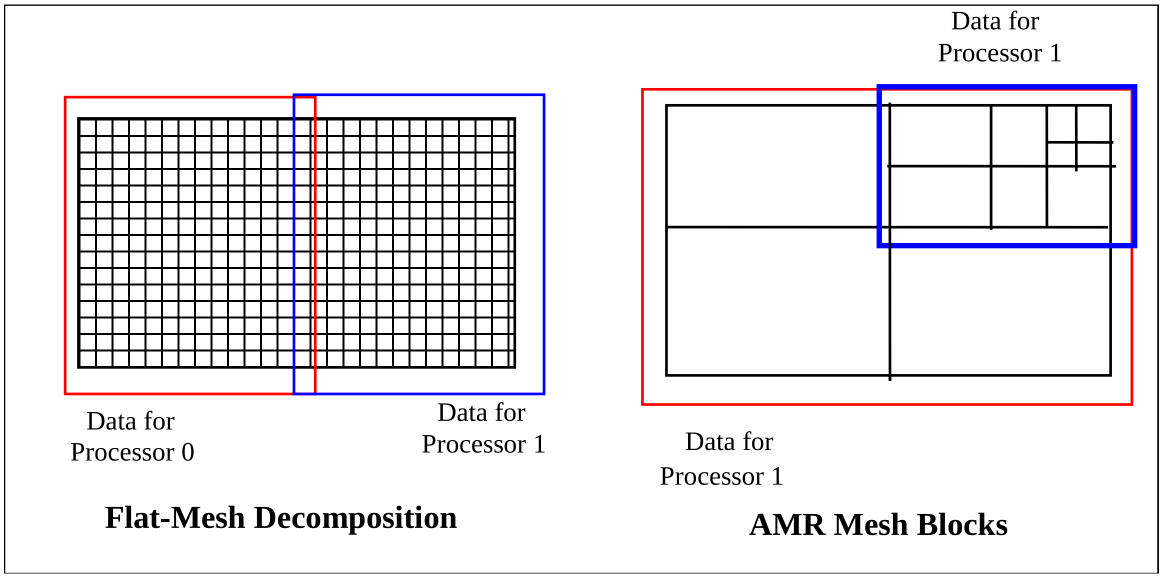

Figure 1. Processor assignment for flat-mesh vs. AMR CTH.

For each material defined in the calculation, CTH calculates a per cell scalar for “volume fraction.” It is the fraction of the volume of the cell that is occupied by that material. For example, if a material’s volume fraction for a cell is 0.51, the cell is 51% filled by that material.

By utilizing this quantity, it is possible to extract surfaces from the entire mesh that correspond to the fragments of BAD. It is the extraction of these isosurfaces that is at the heart of producing the desired mass and velocity information.

Producing realistic fragments from physics-based simulations is an active area of study. Researchers at Lawrence Livermore National Laboratory (*2*) have demonstrated with a Lagrangian code the effectiveness of providing a statistical distribution of fracture properties in simulations. Here the technique is incorporated into the Eulerian code CTH and applied to modeling this ballistic experiment. In a conventional CTH simulation, all cells containing target material have the same set of fracture model parameters, so all fail in the same way. This effect is shown graphically in figure 2a, which shows the bulge on the rear of the target plate just prior to the penetrator breaking through. Damage is shown in this figure by coloring—blue is no damage, red is fully damaged. Notice the uniformity and symmetry of the damage in the bulge. The new model installed in CTH provides a spatially random distribution of values for the initial failure strain (i.e., the failure stain under initial conditions of pressure, temperature, and strain-rate), although in the aggregate, its population is Weibull-distributed. This causes nonuniform, stochastic failure of the armor plate, as shown in figure 2b. The resultant BAD field is strongly dependent on the nature of the Weibull distribution of the fracture parameter, as quantified by the Weibull modulus, which is a user-supplied input to CTH. As an analogy, think of the Weibull modulus as determining the standard deviation of the distribution of the fracture model parameter. A Weibull modulus of 2 provided the results shown in figure 2b. Comparing figures 2a and 2b shows that a much more realistic fragmentation of the target is obtained with the distributed fracture parameter approach.

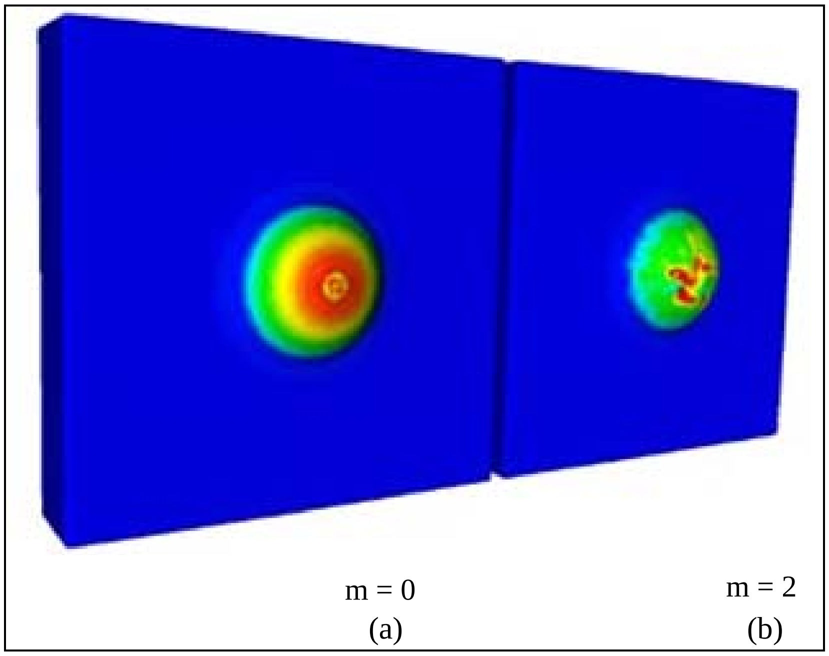

Figure 2. Bulge of the rear of the plate showing damage just prior to breakout (60 μs after impact) for (a) m = 0 and (b) m = 2.

CTH has recently added an analysis facility called Spymaster.  Spymaster is capable of generating images of the calculation during run-time.  It can be accessed during the execution of the original calculation or as a postprocessing step.  Spymaster is designed to generate images, but we have modified it to produce polygonal output of isosurfaces, which can then be further processed to produce the desired information.

Once the isosurfaces have been generated, as in figure 3, the algorithm for extracting mass and velocity of the debris field proceeds as follows:

- Determine individual debris particles by extracting connected regions.
- Determine average X,Y, and Z velocity for each region.
- Determine the volume of each region using an algorithm based on the discrete form of the divergence theorem.
- Multiply by material density to determine mass.
- Sort regions by volume.
- Format output.

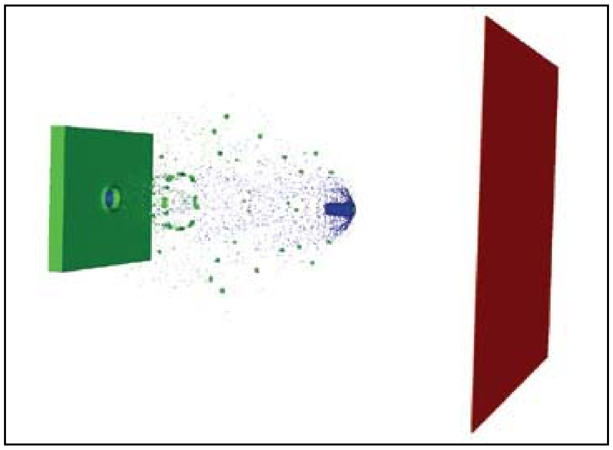

Figure 3.  Isosurface of BAD calculation.

All of these functions are accomplished via a single Python script.  By using the ICE, this script is able to access large chunks of functionality from a relatively small amount of code.  For example, determining the volume of a connected region is accomplished in less than 10 lines of scripting code.

## 3. OnceWare

Developing narrowly targeted, specific applications like this BAD quantifier is a regular occurrence. We refer to these applications as OnceWare. These are applications that are quickly developed, used for very specific applications, and are never intended to be general purpose tools. To facilitate the development of such applications, we take advantage of the ICE.

ICE is a collection of software tools—some open source, some locally developed—that facilitate the development of interdisciplinary applications (*3*). Some of the more notable components are the Python programming language, the Hierarchical Data Format (HDF5), the Visualization Toolkit (VTK), and the eXtensible Data Model and Format (XDMF).

XDMF implements a common data hub for both codes and tools. The common data hub is both a data model and data format. That means the information about the data values and how the data are used are available. Known as the XDMF, the data hub utilizes Extensible Markup Language (XML) and HDF5 to provide a flexible, yet powerful, active data hub (figure 4). In addition to disk files, the physical transfer of data is handled by a distributed shared memory system called Network Distributed Global Memory (NDGM). NDGM provides access to a virtual, contiguous buffer through a client-server architecture (*4*). A widely used HDF5 is used to provide an NDGM buffer with a structure. The common data hub facility provided by HDF5 and NDGM is effectively used to manage data between different software systems and effectively used to coordinate activities between different codes. This enables researchers and engineers to quickly couple production-level parallel high-performance computing codes and tools from different disciplines to assemble complex systems from large chunks of functionality.

Producing XDMF data from CTH requires the addition of commands to Spymaster. This is accomplished by interfacing with the C-like language interpreter, S-Lang, which Spymaster uses to parse input commands. Since this requires additional C code to be added to Spymaster and linkage with the XDMF library, a new executable (IceSpy) is produced that understands all normal Spymaster commands plus additional commands used to produce XDMF.

The most notable of these additional commands is XdmfIsoAllMaterial( scalar, mirror). This command utilizes the Spymaster internal isosurface generator to produce a surface for each material in the calculation where the material volume fraction is 0.51. The scalar is used to interpolate a value such as pressure on the surface, while mirror will reflect the surface across planes of symmetry. Instead of the isosurface generator producing a two-dimensional image, however, this new command produces an XDMF file with the polygons for the surfaces, which can then be imported into EnSight or ParaView.

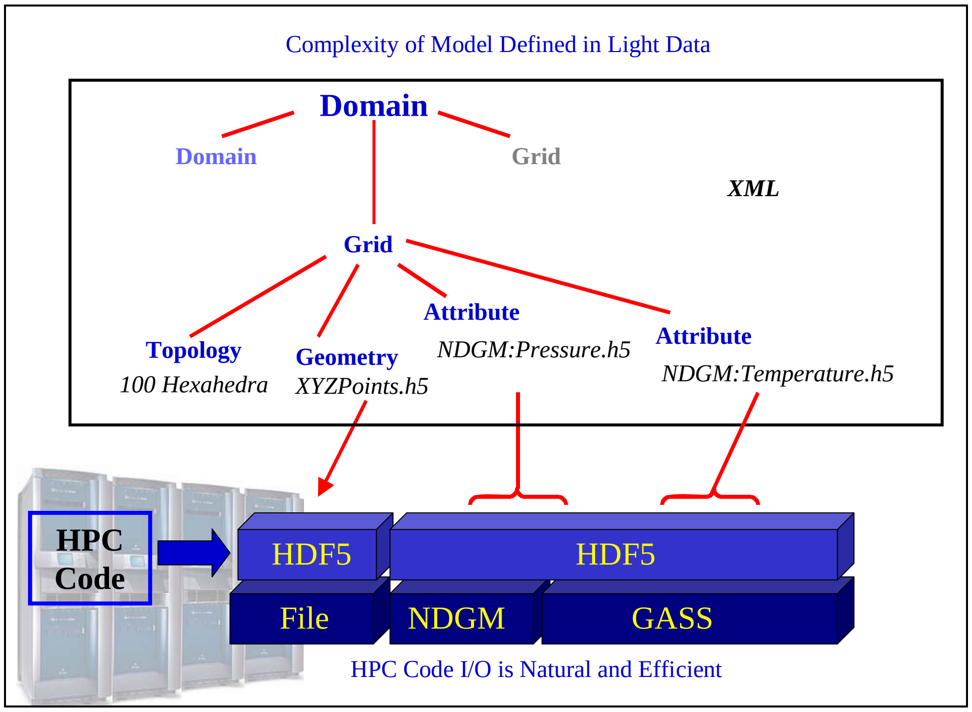

Figure 4. The eXtensible data model and format.

Additional commands allow finer control-like producing surfaces of scalars other than material volume fraction. While useful by themselves, adding additional commands requires recompiling and relinking of the IceSpy executable. An important limitation is that the internal Spymaster isosurface generator will only interpolate one scalar value onto the surface at a time. This means that in order to color the surface by more than one scalar we must take another approach.

## 4. Embedding an Interpreter

Python is a heavily used, interactive, object-oriented, programming scripting language (*5*). Projects like SPaSM (Scalable Parallel Short-range Molecular-dynamics) (*6*) and VTF (Virtual Test Facilty) (*7*) have shown that a Python interpreter can be embedded in a parallel high-performance computing code to provide a flexible interface to a wide variety of functionality. Using this concept, we embedded a Python interpreter into the IceSpy executable. The following two additional commands have been added to IceSpy to access Python from CTH: XdmfPythonExecFile( filename ) and XdmfPythonExec( string ). These commands initialize the Python interpreter and execute Python commands from a file or a string, respectively.

A Python interpreter is fairly thin and has limited functionality beyond the basic language constructs. Additional functionality is obtained by importing external modules. For example, importing the xml.dom module allows Python scripts to easily parse XML documents. The VTK has been wrapped to allow a Python script to access its functionality. Once imported, the VTK module allows the user to create VTK objects and call methods from python; no code needs to be recompiled or linked.

The same has been done for the CTH data. Once in the Python interpreter, the script can access CTH data via classes and methods that are loaded from a module. Methods to retrieve saved variable names and values are provided. One of the most important of these methods generates a VTK rectilinear grid from a CTH block (AMR or flat). By importing VTK functionality, the script can then use the full power of the VTK system (*8*) to perform a myriad of visualization functions on the CTH data. Since the compute-intensive portion of this processing is accomplished in the underlying C or C++ code, Python actually adds very little computational overhead while providing enormous flexibility.

## 5. Connectivity and Volume

Once we have polygonal isosurfaces of the debris field, the next step is to determine the individual particles. The first step in this process is to merge points in the triangular mesh that are within a certain tolerance. This is necessary since the isosurfaces were generated in parallel; coincident points may have been generated by two or more processors.

To determine the individual particles, we start with the first triangle and find all of the triangles that share one or more nodes. We continue finding connected triangles, removing these from the candidate list. Once there are no more triangles that share nodes, the group of triangles is considered to be a discrete object, and an ordinal scalar value for region number is added to those triangles.

Determining the average velocity of this region is trivial, but the same cannot be said for determining the volume. To find the volume, we use an algorithm based on the discrete form of the divergence theorem, which is described by Alyassin et al. (*9*). It is implemented in the VTK filter, vtkMassProperties, which is part of the ICE environment.

Of all of the factors that can limit the accuracy of the volume, the most significant is the original resolution of the CTH mesh. Consider a perfect sphere—without proper resolution, the resulting faceted shape may significantly underpredict the particle volume. So far, this has not been a major problem. The total mass of all of the particles is determined at each iteration. The computed mass is generally within 1% of the initial (t = 0) mass.

As the final output is formatted for input to other codes, particles below a user-determined threshold are filtered out. For every iteration, there is one line per particle that lists its centroid, volume, mass, and velocity.

As shown in figure 5, the shape of the distribution curves between experiment and simulation are similar. Work further continues to match the actual masses. This work has shown that

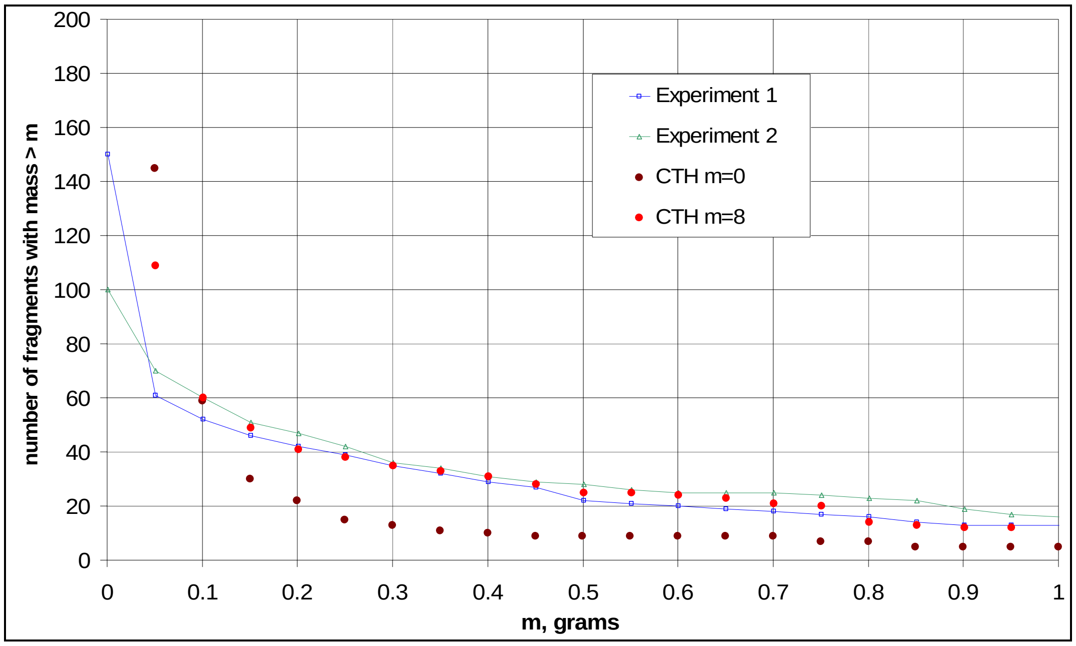

Figure 5. CTH prediction of mass distribution of fragments compared to experimental results. Here the statistical fracture technique is applied to only the target. This figure indicates that with the proper choice of Weibull modulus, and with the statistical model applied to both the target and the penetrator, a more realistic debris field can be obtained than that which arises from the classic method of using a constant parameter (m = 0).

simulating BAD experiment is now within the ability of current assets, and simulations can be successfully exploited to supplement the expensive experiments. Furthermore, the new capabilities of statistical fracture and automatic fragment quantification make the technique more useful.

## 6. Conclusion

This system has been used to quantify the BAD field calculated from a CTH simulation. This is a practical system to accomplish this task in a fairly general fashion. We are able to access large chunks of functionality from a high-level scripting language with a small amount of code through the use of ICE. The fragment extraction tools within ICE are one important piece in the process of using physics-based simulations to improve vehicle design.

We are currently working to improve the performance, ease of use, and accuracy of the process and to improve the systems interoperability with external codes used to assess vehicle vulnerability and personnel risk.

## 7. References

1. Littlefield, D. L. *A Brief Description of New Algorithms Incorporated Into CTH: A Model for Rigid Obstacles and Interface for Coupling With Structural Codes*; Texas Institute for Computational and Applied Mathematics, The University of Texas: Austin, TX, November 2001.

2. Springer, H. K.; Becker, R. C.; Couch, R. *Fragmentation Modeling of Ductile [sic] With Heterogeneous Microstructure, Tri-Lab Engineering Conference*; UCRL-JC-154019-ABS; Lawrence Livermore National Laboratory: Santa Fe, NM, 2003.

3. Clarke, J. A.; Namburu, R. R. A Distributed Computing Environment for Interdisciplinary Applications. *Concurrency and Computation: Practice and Experience* **2002**, *14* (13–15), 1161–1174.

4. Clarke, J. Emulating Shared Memory to Simplify Distributed Memory Programming. *IEEE Computational Science and Engineering* **1997**, *4* (1), 55–62.

5. Python Programming Language – Official Website. http://www.python.org (accessed June 2006).

6. Beazley, D. M.; Lomdahl, P. S. Feeding a Large-Scale Physics Application to Python. *Proceedings of the 6th International Python Conference*, San Jose, CA, October 1997, pp 21–29.

7. Cummings, J.; Aivazis, M.; Samtaney, R.; Radovitzky, R.; Mauch, S.; Meiron, D. A Virtual Test Facility for the Simulation of Dynamic Response in Materials. *The Journal of Supercomputing* **2002**, *23* (1), 39–50.

8. Schroeder, W.; Martin, K.; Lorensen, B. *The Visualization Toolkit, Third Edition*; Kitware Inc.: Clifton Park, NY, 2002.

9. Alyassin, A. M.; Lancaster, J. L.; Downs, J. H. III; Fox, P. T. Evaluation of New Algorithms for the Interactive Measurement of Surface Area and Volume. *Med Phys.* **1994**, *21* (6).

| NO. OF COPIES | ORGANIZATION |
|---|---|
| 1 (PDF ONLY) | DEFENSE TECHNICAL INFORMATION CTR<br>DTIC OCA<br>8725 JOHN J KINGMAN RD<br>STE 0944<br>FORT BELVOIR VA 22060-6218 |
| 1 | US ARMY RSRCH DEV & ENGRG CMD<br>SYSTEMS OF SYSTEMS INTEGRATION<br>AMSRD SS T<br>6000 6TH ST STE 100<br>FORT BELVOIR VA 22060-5608 |
| 1 | DIRECTOR<br>US ARMY RESEARCH LAB<br>IMNE ALC IMS<br>2800 POWDER MILL RD<br>ADELPHI MD 20783-1197 |
| 3 | DIRECTOR<br>US ARMY RESEARCH LAB<br>AMSRD ARL CI OK TL<br>2800 POWDER MILL RD<br>ADELPHI MD 20783-1197 |
| | ABERDEEN PROVING GROUND |
| 1 | DIR USARL<br>AMSRD ARL CI OK TP (BLDG 4600) |

|  |  |
|---|---|
| NO. OF COPIES | ORGANIZATION |
|  | ABERDEEN PROVING GROUND |
| 1 | DIR USARL<br>AMSRD ARL CI HC<br>R NAMBURU |

INTENTIONALLY LEFT BLANK.